CS1337 – U 16

Homework #3-2

Assigned June 22, due June 30 at 11:59 PM

This homework assignment gives you the opportunity to practice structures, pointers to structures, arrays of pointers to structures, dynamic allocation and file input/output.

HW3-2 (Graded out of 100)

Write a program that uses a structure to store grocery product information. For each product we store the following information in a structure:

- PLU code: Product Look Up Code. Unique for each product, stored as a string.
- Product Name,
- Product Sales Type: 0 = per unit, 1 = per pound
- Price per Pound or price per unit,
- Current Inventory Level.

The structures are accessed through an array of pointers, where each pointer points to a structure.

Your program should:
Read grocery product information from a file, and populate the structures. Below is an example of file content. Each line in the file corresponds to one product.

```
A001 Apples 1 0.90 21
A002 Peaches 1 0.82 11
A006 Avocados 0 1.54 27
A008 Mangos 0 1.69 19
```

Then the program should display a menu for the user to choose from:
1 – Checkout
2 – Close and exit

<u>Checkout</u>
If the user chooses checkout, the user can purchase multiple products in each checkout. The user is asked to enter the PLU code, then the quantity (weight or # of units, depending on the sales type) for each product. The program keeps up with the total cost ($). The user can enter 0 for a PLU to indicate the end of checkout. The program then displays the total price, and displays the menu again. The program should do input validation (PLU can be only letters and digits, and quantity must be positive). Your program should make sure the quantity bought is not more than the inventory level. For example, if the user wants to buy 10 apples, but only 5 apples are in the inventory, the program will sell only 5 apples. The inventory level should be automatically updated with each purchase.

<u>Close and exit</u>
If the user chooses this option, all the updated product information should be displayed and written into the file, then the program exits.

1. Requirements

a) Structure definition

```
struct Product
{
   string PLU;
   string name;
   int salesType;
   double unitPrice;
   double inventoryLevel;
};
```

b) Array definition

```
const int SIZE = 100; // Assume there are at most 100 products
```

Define an array of pointers to `Product`, of size `SIZE`, locally in `main`.

c) Functions

You are required to implement your program with at least 3 functions. You are encouraged to implement more functions to make your program more modular.

To read a file and populate the structures, call a function with the following prototype and description:

```
/* This function takes as arguments a file name and an array of
pointers to Product, along with
the max size of the array. It opens the file and reads the inventory
from the file.
It reads line by line, where each line corresponds to a product.
For each product, it dynamically allocates a Product structure and
assigns the pointer to an element of the array.
It does input validation (PLU has only letters or digits, sales type
can be only 0 or 1,
unit price must be > 0, inventory level >= 0) and populates the
structure with the data entered.
The structure content is displayed. The function terminates when there
is no more data in the file
or when the array is full. It also terminates upon error condition.
It returns the number of products if all the data was valid, -1 in all
other cases of error.
*/
int readInventory(string fName, Product * arrayProd[], int max_size);
```

For the checkout, implement a function with the following prototype and description:

```
/* This function does the checkout. It takes as argument the array of
pointers to Product, and the number of products in the
inventory. It reads the PLU and quantity from the user, updates the
data in the array to reflect the user's purchase,
does input validation on PLU and purchased quantity (> 0), prompts the
user for more pur0chases.
```

```
The user types a PLU of zero when done. The function returns thenProd
total purchase price.
*/
double checkout(Product * arrayProd[], int nProd);
```

For the close and exit, implement a function with the following prototype and description:

```
/* This function reads the structures and writes the data into a file.
The function takes as arguments the file name, the array of pointers
to Product and the number of
products in the inventory. Returns true if sucessful, false otherwise.
*/
bool updateInventory(string fName, Product * arrayProd[], int nProd);
```

d) Outline of main

To demonstrate your program, write a main function with the following outline. The files `“productData0.txt”`, `“productData1.txt”`, `“productData2.txt”`, `“productData3.txt”` and `“productData4.txt”` have been intentionally corrupted to test your input validation, while the file `“productData.txt”` is properly formatted. All the files are on eLearning.

```
Call readInventory for file “productData0.txt” // The program is expected
to display an error
Call readInventory for file “productData1.txt” // The program is expected
to display an error
Call readInventory for file “productData2.txt” // The program is expected
to display an error
Call readInventory for file “productData3.txt” // The program is expected
to display an error
Call readInventory for file “productData4.txt” // The program is expected
to display an error
Call readInventory for file “productData.txt” // No error message is
expected, and the content of all the structures is displayed

Display menu and get user’s choice
While (user’s choice is not Close and exit)
     Call checkout
End while
Call updateInventory to write into file “updatedProductData.txt”
```

You are allowed to hard code all the file names.

e) Style

Make sure you follow the style requirements, especially regarding the comment header for functions, to avoid losing points.

2. Suggestions for Implementation

Apply the “divide and conquer” strategy by splitting the original problem into smaller pieces, and solve one piece at a time.

a) PLU handling

To validate that the PLU consists only of letters and digits, read the PLU as a string, and reuse the same function from HW3-1.

```
/* This function takes a string as argument and returns true if all
the characters
in the string are letters or digits. Else it returns false
*/
bool validatePLU(const string & s);
```

The user can quit by entering a PLU of value zero, but the PLU is read as a string, so you need to convert the PLU read into a numerical value and test for equality with zero.
To convert a string into an `int`, you can use the same `myStoi` function as in HW3-1.

```
/* This function checks the string is not empty and the string has
only digits
before it calls stoi. Returns true if conversion to int was
successful, false otherwise.
The int converted from the string by stoi, if conversion was
successful, is the reference variable res
*/
bool myStoi(const string & s, int &res);
```

b) Read from file

Because the file content can be incorrectly formatted, to read from the file, you need to use `getline` to read the whole line into a string. Then you can reuse `extractWord` function from HW2-2 to extract the data items, and convert them one by one to the proper data type, with input validation.

Sales type input validation

To convert the sales type to an `int`, use the same `myStoi` as in HW3-1.

Price and inventory level input validation

You can write a similar function for `double`, to read the price and inventory level for input validation.

```
/* This function checks the string is not empty and the string has
only digits and at most one decimal point
before it calls stod. Returns true if conversion to double was
successful, false otherwise.
The double converted from the string by stod, if conversion was
successful, is the reference variable res
*/
bool myStod(const string &s, double &res);
```

c) Checkout

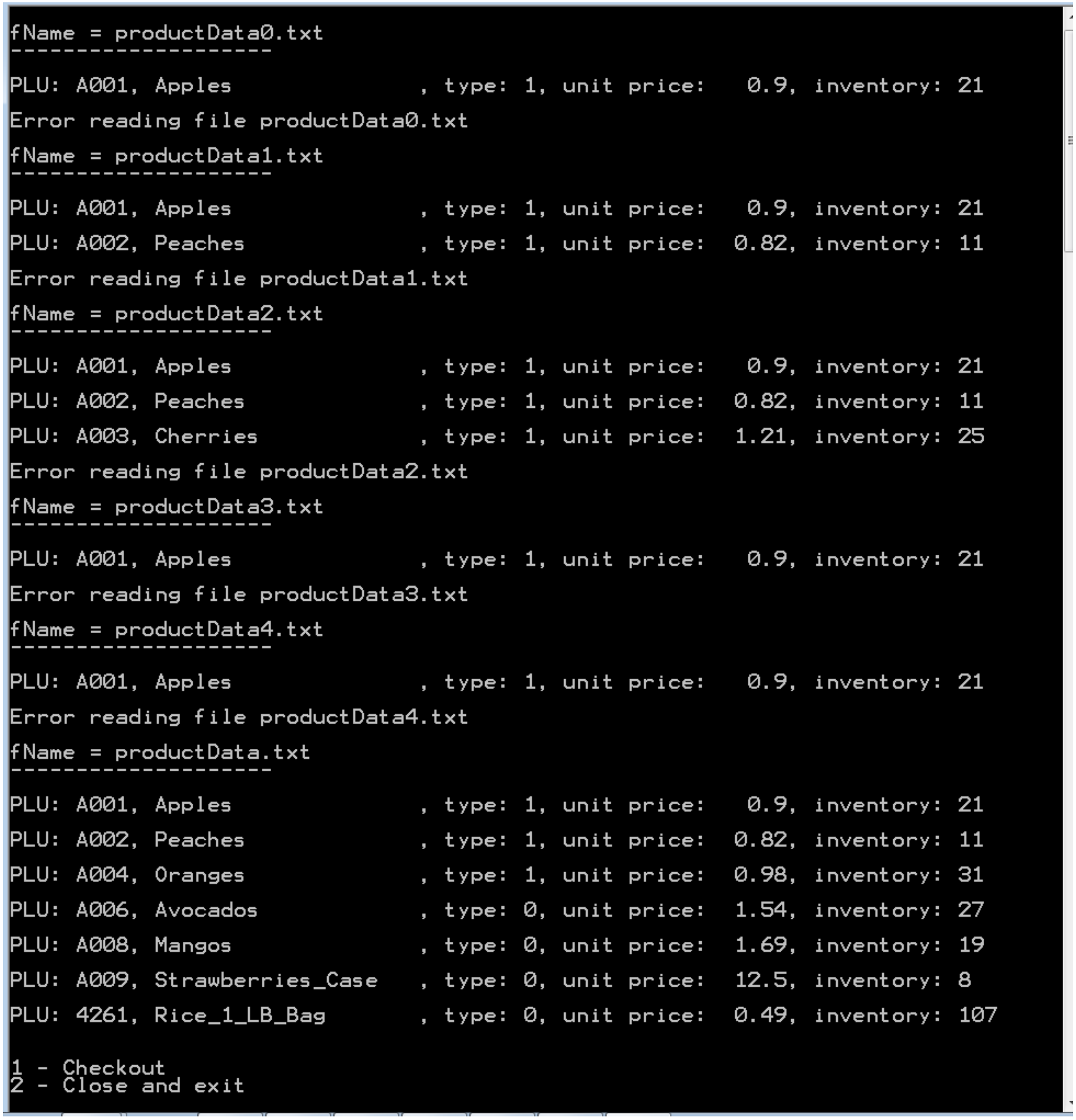
You will need to search for a product, and you can reuse the `searchProduct` function from HW3-1.

```
/* This function takes as argument a PLU string, the array of pointers
to Product and the number of products.
It searches the arrayProducts for a matching PLU
```

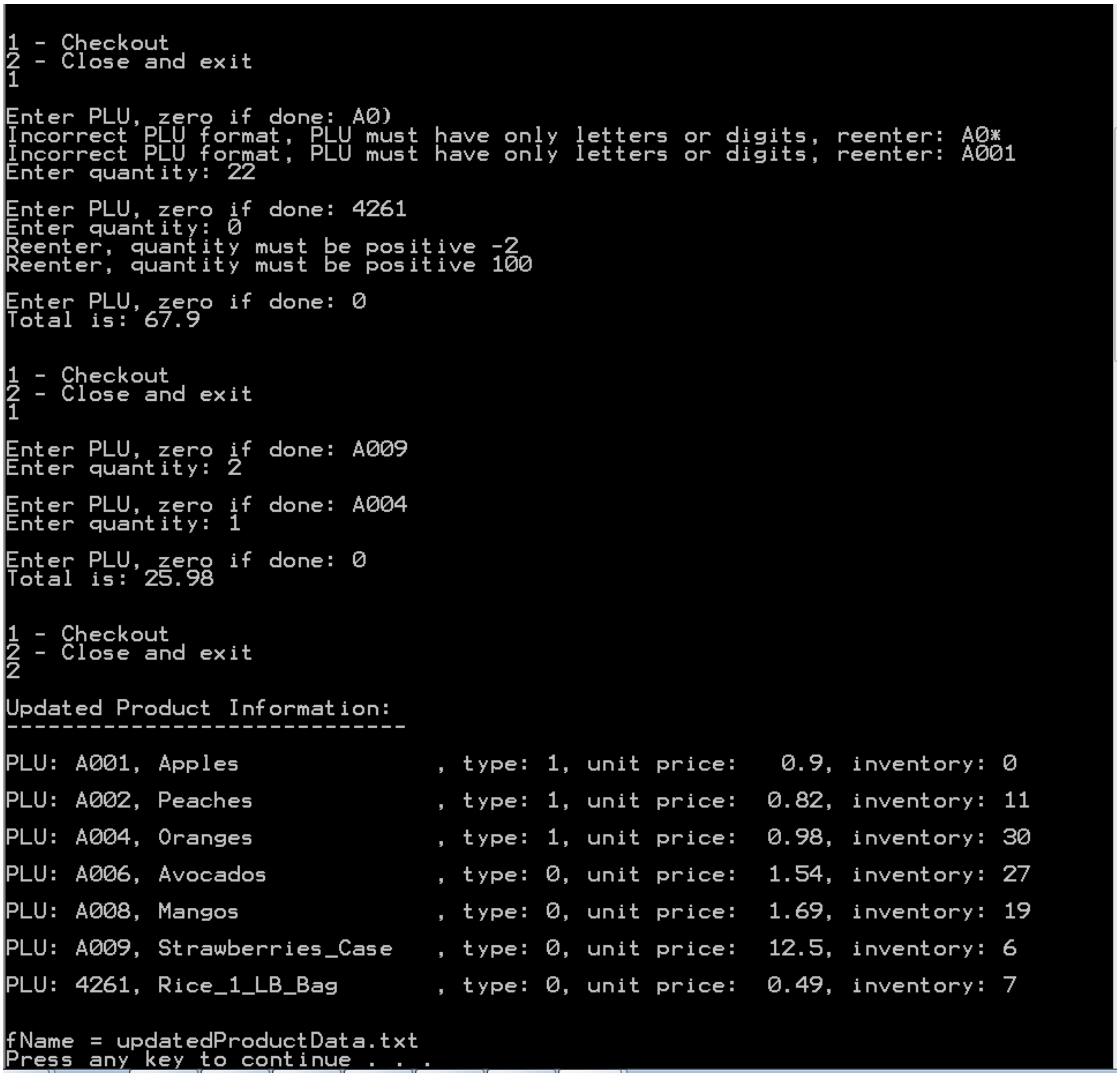
```
and returns the array index if found, or -1 if not found
*/
int searchProduct(const string & pluStr, Product *arrayProd[], int
numProducts);
```

3. Expected Output

Below are two examples of output if your implementation is correct:

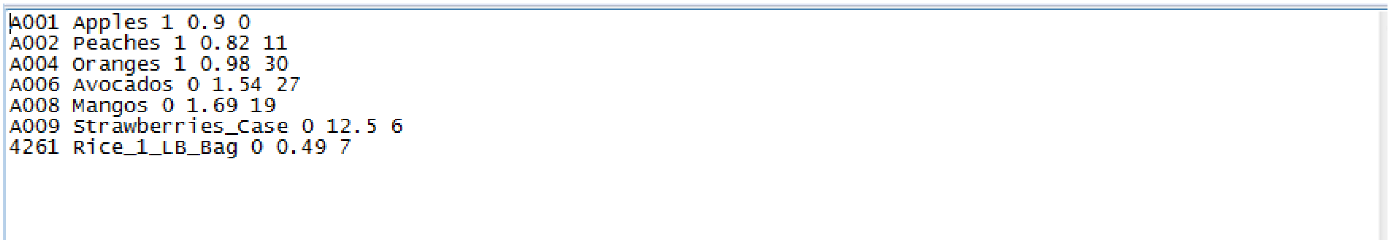


```
fName = productData0.txt
--------------------

PLU: A001, Apples              , type: 1, unit price:   0.9, inventory: 21
Error reading file productData0.txt
fName = productData1.txt
--------------------

PLU: A001, Apples              , type: 1, unit price:   0.9, inventory: 21
PLU: A002, Peaches             , type: 1, unit price:  0.82, inventory: 11
Error reading file productData1.txt
fName = productData2.txt
--------------------

PLU: A001, Apples              , type: 1, unit price:   0.9, inventory: 21
PLU: A002, Peaches             , type: 1, unit price:  0.82, inventory: 11
PLU: A003, Cherries            , type: 1, unit price:  1.21, inventory: 25
Error reading file productData2.txt
fName = productData3.txt
--------------------

PLU: A001, Apples              , type: 1, unit price:   0.9, inventory: 21
Error reading file productData3.txt
fName = productData4.txt
--------------------

PLU: A001, Apples              , type: 1, unit price:   0.9, inventory: 21
Error reading file productData4.txt
fName = productData.txt
--------------------

PLU: A001, Apples              , type: 1, unit price:   0.9, inventory: 21
PLU: A002, Peaches             , type: 1, unit price:  0.82, inventory: 11
PLU: A004, Oranges             , type: 1, unit price:  0.98, inventory: 31
PLU: A006, Avocados            , type: 0, unit price:  1.54, inventory: 27
PLU: A008, Mangos              , type: 0, unit price:  1.69, inventory: 19
PLU: A009, Strawberries_Case   , type: 0, unit price:  12.5, inventory: 8
PLU: 4261, Rice_1_LB_Bag       , type: 0, unit price:  0.49, inventory: 107

1 - Checkout
2 - Close and exit
```



```
1 - Checkout
2 - Close and exit
1

Enter PLU, zero if done: A0)
Incorrect PLU format, PLU must have only letters or digits, reenter: A0*
Incorrect PLU format, PLU must have only letters or digits, reenter: A001
Enter quantity: 22

Enter PLU, zero if done: 4261
Enter quantity: 0
Reenter, quantity must be positive -2
Reenter, quantity must be positive 100

Enter PLU, zero if done: 0
Total is: 67.9


1 - Checkout
2 - Close and exit
1

Enter PLU, zero if done: A009
Enter quantity: 2

Enter PLU, zero if done: A004
Enter quantity: 1

Enter PLU, zero if done: 0
Total is: 25.98


1 - Checkout
2 - Close and exit
2

Updated Product Information:
-----------------------------

PLU: A001, Apples                , type: 1, unit price:   0.9, inventory: 0

PLU: A002, Peaches               , type: 1, unit price:  0.82, inventory: 11

PLU: A004, Oranges               , type: 1, unit price:  0.98, inventory: 30

PLU: A006, Avocados              , type: 0, unit price:  1.54, inventory: 27

PLU: A008, Mangos                , type: 0, unit price:  1.69, inventory: 19

PLU: A009, Strawberries_Case     , type: 0, unit price:  12.5, inventory: 6

PLU: 4261, Rice_1_LB_Bag         , type: 0, unit price:  0.49, inventory: 7

fName = updatedProductData.txt
Press any key to continue . . .
```

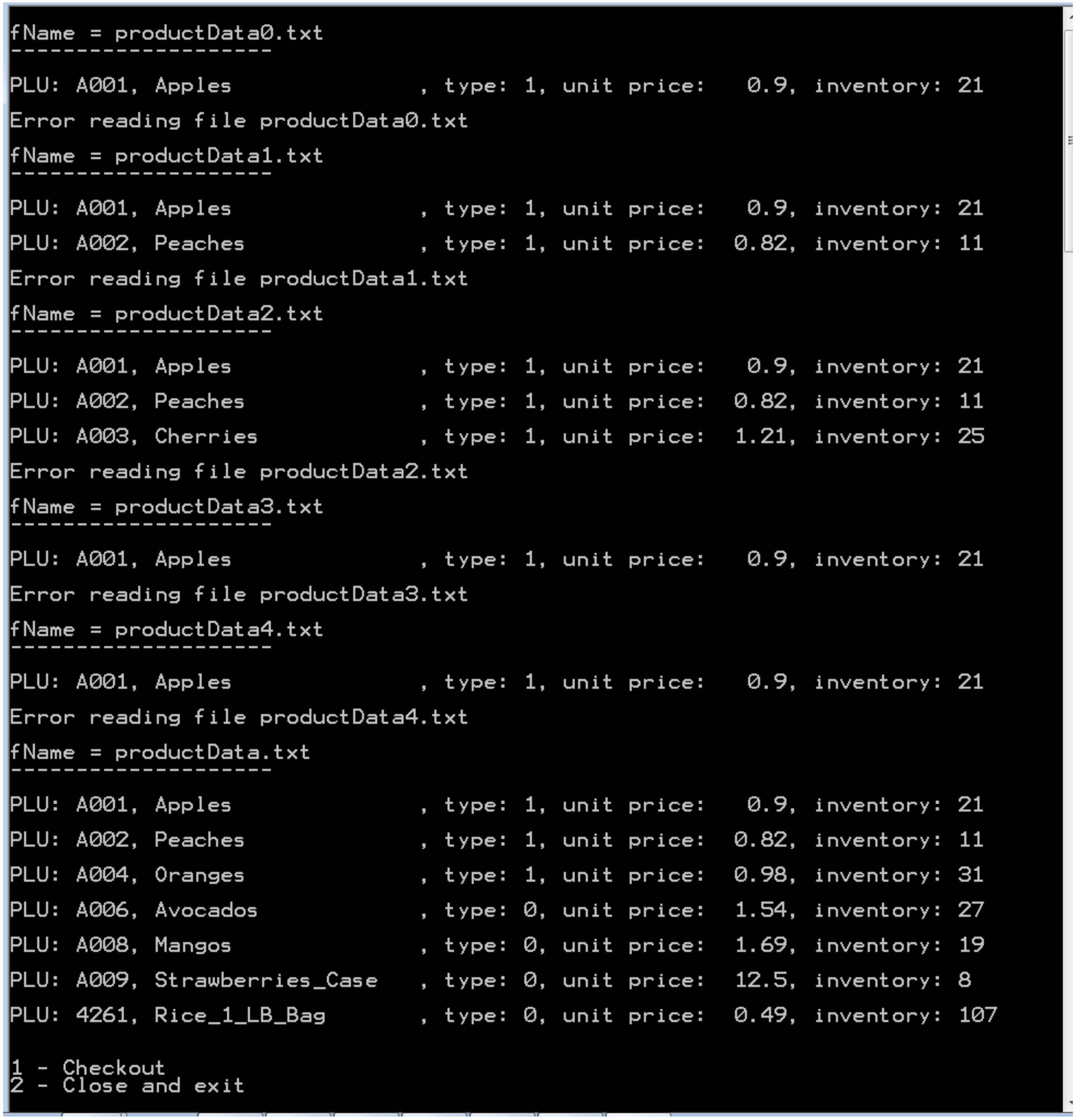
After close and exit, the content of file "`updatedProductData.txt`" should be:



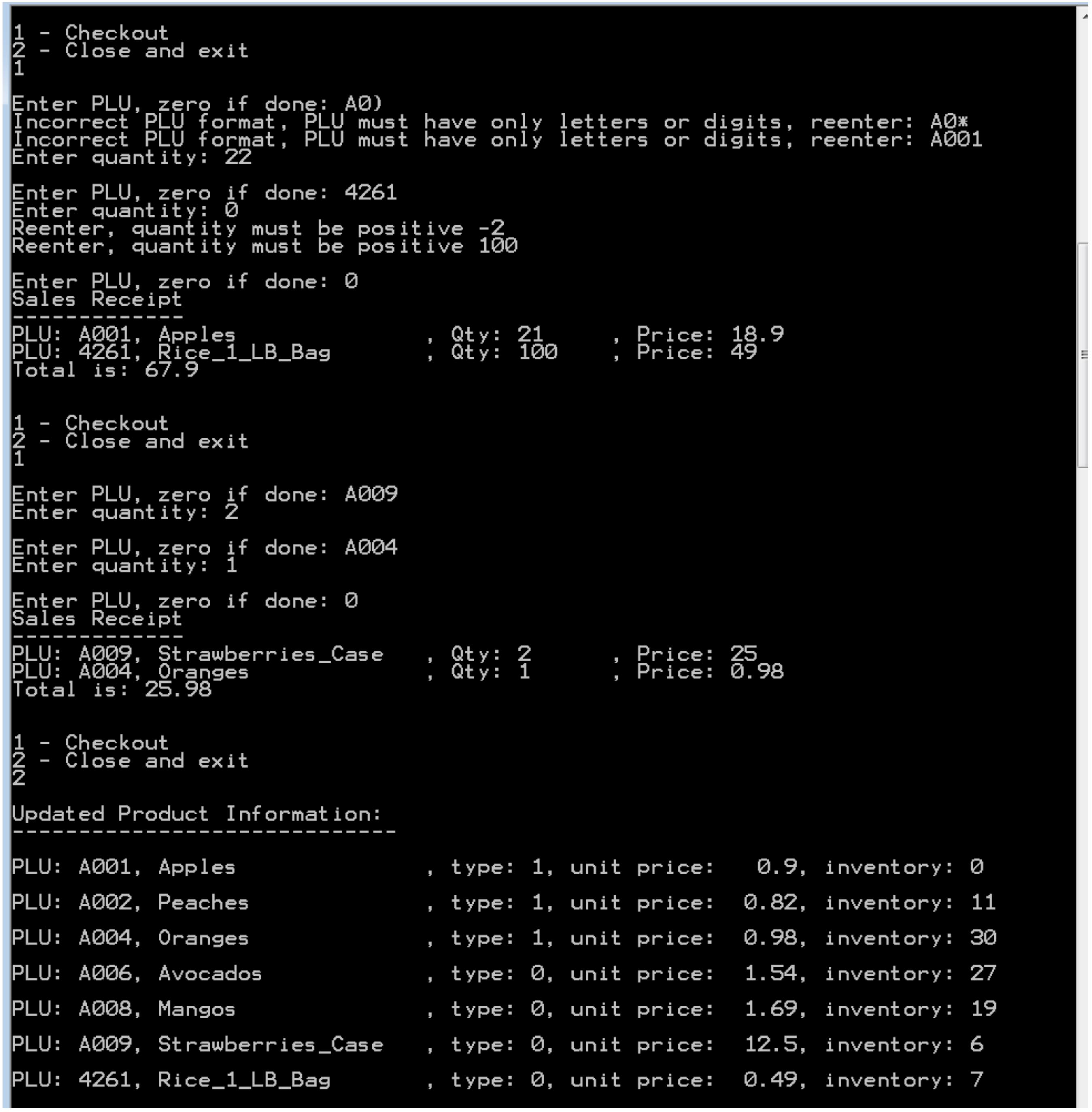
```
A001 Apples 1 0.9 0
A002 Peaches 1 0.82 11
A004 Oranges 1 0.98 30
A006 Avocados 0 1.54 27
A008 Mangos 0 1.69 19
A009 Strawberries_Case 0 12.5 6
4261 Rice_1_LB_Bag 0 0.49 7
```

4. Extra Credit

You can earn up to 10 points extra credit if, in addition to the above, your program prints a list of items and their prices, along with the total price, at the end of each checkout. This is the equivalent of a sales receipt.



```
fName = productData0.txt
--------------------

PLU: A001, Apples                , type: 1, unit price:   0.9, inventory: 21
Error reading file productData0.txt
fName = productData1.txt
--------------------

PLU: A001, Apples                , type: 1, unit price:   0.9, inventory: 21
PLU: A002, Peaches               , type: 1, unit price:  0.82, inventory: 11
Error reading file productData1.txt
fName = productData2.txt
--------------------

PLU: A001, Apples                , type: 1, unit price:   0.9, inventory: 21
PLU: A002, Peaches               , type: 1, unit price:  0.82, inventory: 11
PLU: A003, Cherries              , type: 1, unit price:  1.21, inventory: 25
Error reading file productData2.txt
fName = productData3.txt
--------------------

PLU: A001, Apples                , type: 1, unit price:   0.9, inventory: 21
Error reading file productData3.txt
fName = productData4.txt
--------------------

PLU: A001, Apples                , type: 1, unit price:   0.9, inventory: 21
Error reading file productData4.txt
fName = productData.txt
--------------------

PLU: A001, Apples                , type: 1, unit price:   0.9, inventory: 21
PLU: A002, Peaches               , type: 1, unit price:  0.82, inventory: 11
PLU: A004, Oranges               , type: 1, unit price:  0.98, inventory: 31
PLU: A006, Avocados              , type: 0, unit price:  1.54, inventory: 27
PLU: A008, Mangos                , type: 0, unit price:  1.69, inventory: 19
PLU: A009, Strawberries_Case     , type: 0, unit price:  12.5, inventory: 8
PLU: 4261, Rice_1_LB_Bag         , type: 0, unit price:  0.49, inventory: 107

1 - Checkout
2 - Close and exit
```



```
1 - Checkout
2 - Close and exit
1

Enter PLU, zero if done: A0)
Incorrect PLU format, PLU must have only letters or digits, reenter: A0*
Incorrect PLU format, PLU must have only letters or digits, reenter: A001
Enter quantity: 22

Enter PLU, zero if done: 4261
Enter quantity: 0
Reenter, quantity must be positive -2
Reenter, quantity must be positive 100

Enter PLU, zero if done: 0
Sales Receipt
-------------
PLU: A001, Apples              , Qty: 21     , Price: 18.9
PLU: 4261, Rice_1_LB_Bag       , Qty: 100    , Price: 49
Total is: 67.9


1 - Checkout
2 - Close and exit
1

Enter PLU, zero if done: A009
Enter quantity: 2

Enter PLU, zero if done: A004
Enter quantity: 1

Enter PLU, zero if done: 0
Sales Receipt
-------------
PLU: A009, Strawberries_Case   , Qty: 2      , Price: 25
PLU: A004, Oranges             , Qty: 1      , Price: 0.98
Total is: 25.98


1 - Checkout
2 - Close and exit
2

Updated Product Information:
-----------------------------

PLU: A001, Apples              , type: 1, unit price:   0.9, inventory: 0

PLU: A002, Peaches             , type: 1, unit price:  0.82, inventory: 11

PLU: A004, Oranges             , type: 1, unit price:  0.98, inventory: 30

PLU: A006, Avocados            , type: 0, unit price:  1.54, inventory: 27

PLU: A008, Mangos              , type: 0, unit price:  1.69, inventory: 19

PLU: A009, Strawberries_Case   , type: 0, unit price:  12.5, inventory: 6

PLU: 4261, Rice_1_LB_Bag       , type: 0, unit price:  0.49, inventory: 7
```